1	6	
2	6	
3	6	
4	6	
5	6	
6	10	
∑	40	

Logikai tervezés vizsga

2012. 01. 11.

Név: …….. NEPTUN kód: ……..
Email:

A feladatokat önállóan, meg nem engedett segédeszközök használata nélkül oldottam meg:

……

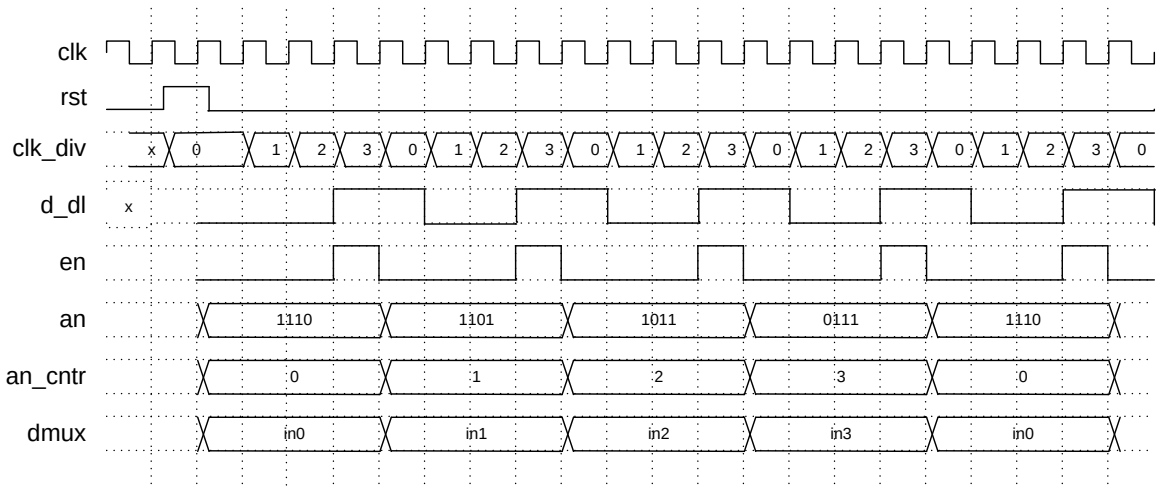
1. Ismertesse a Xilinx Spartan-3 CLB felépítését (milyen erőforrások találhatók a CLB-ben, ezek hogyan használhatók).
   Lásd: slide-ok.
   De a lényeg: 1 CLB = 4 SLICE. A SLICE-ban 2 4 bementű LUT, 2 tároló (latch-ként vagy FF-ként használható), carry logika és dedikált mux található. A SLICEM LUT-jai meg tudnak valósítani 16 mély 1 bit széles shift regisztert (nem resetelhető) és ugyanekkora aszinkron olvasású, szinkron írású memóriát.
2. Ismertesse a különböző Xilinx FPGA típusokban található jelfeldolgozó elemeket (szorzó, DSP modul)!
   MUL18x18, MUL18x18S (kimeneti reg), MUL18x18SIO (ki és bemeneti reg)
   DSP48: 18x18 mul + 48 accu
   DSP48E: 25x18 mul, 48 accu, logikai műveletek
   DSP48E1: DSP48E + preadder
   DSP48A: ~DSP48E, csak 18x18 mul
3. Mi a szerepe a JTAG interfésznek FPGA-k esetén? Milyen további támogatást ad az interfész áramkörök debug-olásra?
   - konfiguráció
   - boundary scan
   - kommunikáció logikai analizátorral (Chipscope)
   - (mikroprocesszoros rendszer debug)
4. Mit jelent a „source synchronous, edge aligned” interfész (rajzoljon időzítési diagramot)? Hogyan lehet egy ilyen interfész illesztését megoldani Virtex-6 FPGA-ban (mind a kimeneti, mind pedig a bemeneti irány lehetőségeit ismertesse)?
   - az adatot küldő egység szolgáltatja az átvitelt ütemező órajelet (source sync)
   - az adatváltás időpontja egybeesik az órajel éllel (edge aligned)
   - ebből következően közvetlenül ez az órajel nem használható mintavételezésre
     - nem DDR esetén az órajel negáltjára lehet mintavételezni

    - DDR esetén 90 fokkal el kell tolni a bejövő órajelet, ez történhet DCM-mel (ha szabadon futó az órajel) vagy IODELAY-jel (szabadon futó és nem szabadon futó esetben is jó)
- a bejövő adatot IO regiszterbe mintavételezzük
- kimenet: mind a kimeneti órajel, mind pedig az adat DDR kimeneti FF-on keresztül kerül az FPGA lábra

5. Adott az alábbi Verilog kód.

```
module seg7_cntrl(
    input            clk, rst,
    input      [3:0] din0, din1, din2, din3,
    output reg [3:0] dmux,
    output reg [3:0] an
);

reg [1:0] clk_div;
always @ (posedge clk, posedge rst)
if (rst)    clk_div <= 0;
else        clk_div <= clk_div + 1;

reg d_dl;
always @ (posedge clk)
    d_dl <= clk_div[1];

reg en;
always @ (posedge clk)
if (rst)    en <= 0;
else        en <= (clk_div[1] & ~d_dl);

reg [1:0] an_cntr;
always @ (posedge clk)
if (rst)
begin
       an      <= 4'b1110; an_cntr <= 0;
end
else if (en)
begin
       an      <= {an[2:0], an[3]};  an_cntr <= an_cntr + 1;
end
always @ ( * )
case (an_cntr)
    0: dmux <= din0;
    1: dmux <= din1;
    2: dmux <= din2;
    3: dmux <= din3;
endcase
endmodule
```

A kód alapján egészítse ki az alábbi hullámformát: rajzolja be, hogy viselkedési szimuláció esetén mit látnánk a hullámforma ablakban. A jelváltások időpontját pontosan jelölje függőleges vonalak segítségével. A 0. pillanatban minden jel értéke „nem definiált”.



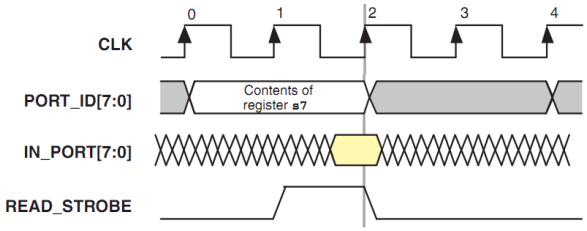
6. Picoblaze processzorhoz szeretnénk illeszteni egy 8 bites, kétirányú egyszerű I/O perifériát. A periféria az alábbi regiszterekkel rendelkezzen:

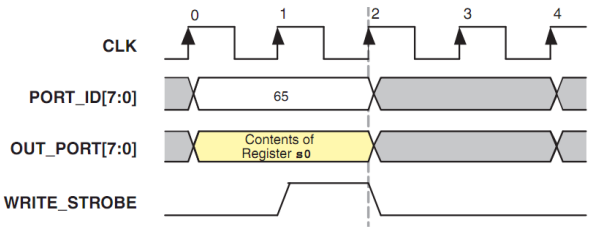
Cím	**WR/RD**	
0x0	WR/RD	Irány regiszter (bitenként). 0: az adott bit kimenetkén funkcionál. 1: az adott bit bemenetként funkcionál.
0x1	WR	Kimeneti adat.
0x1	RD	Az I/O porton elérhető adat.

A megvalósítandó modul portjai:

clk	I	skalár	A Picoblaze órajele.
port_id	I	[7:0]	Picoblaze címvonal.
write_strobe	I	skalár	Picoblaze írás engedélyező vonal.
read_strobe	I	skalár	Picoblaze olvasás engedélyező vonal.
out_port	I	[7:0]	A Picoblaze kimeneti adatbusza.
in_port	O	[7:0]	A Picoblaze bemeneti adatbusza.
io_pin	I/O	[7:0]	Az FPGA 8 I/O lábára csatlakozó adatvonal, melynek állapotát a periféria kezeli.

Adja meg a szükséges funkcionalítást megvalósító modul Verilog vagy VHDL kódját. A Picoblaze buszciklusának időzítési diagramját az alábbi ábra mutatja.





```
module io_if(
	input clk, write_strobe, read_strobe,
	input [7:0] port_id, out_port,
	output [7:0] in_port,
	inout [7:0] io_pin);

// ha a picoblaze a 0x0 címre ír, akkor az irányregisztert írja
// az írás engedélyező jele a write_strobe
reg [7:0] dir, data_out;
always @ (posedge clk)
if (write_strobe==1 & port_id==0)
	dir <= out_port;

// 0x1 címen az adatregiszter
always @ (posedge clk)
if (write_strobe==1 & port_id==1)
	data_out <= out_port;

// olvasási multiplexer
// nem használja a read_strobe-t, az egyetlen kritérium (lásd diagram), hogy a read_strobe 1 értéke
// alatt a kimeneti busz az olvasott adatot tartalmazza
// 0x0 címen az irány regiszter,
// 0x1 címen az I/O port olvasható vissza
reg [7:0] rd_mux;
always @ ( * )
if (port_id==0)
	rd_mux <= dir;
else if (port_id==1)
	rd_mux <= io_pin;
else
	rd_mux <= 8'bxxxxxxxx;

// ha az irány regiszter i-ik bitje 0, akkor meghajtjuk az adott I/O lábat, egyébként pedig
// nagyimpedanciás állapotba rakjuk, azaz kívülről hajtható meg
genvar i;
generate
	for (i=0; i<8; i=i+1)
	begin: drv_io_pin
		assign io_pin[i] = (dir[i]==0) ? data_out[i] : 1'bz;
	end
endgenerate

endmodule
```